**John Arena**

**Professor Gertner**

**CSC 342/343**

**Lab 1**

**Due 2/20/19**

**Spring 2019**

**Table of Contents**

**Note: Files sent to your email address on 12/3/18**

**Section 1) Objective pg. 3**

**Section 2) Description and Specification pg. 3**

**Section 3) Simulations pg. 12**

**Section 4) Conclusion pg. 26**

**Section 5) Appendix pg. 48**

**Section 6) Electric Layout pg. 55**

**Section 7) LTSPICE for Electric Layout pg. 71**

**Section 8) IRSIM for Electric Layout pg. 89**

**Section 9) Summary of Measurements pg. 96**

**Section 10) LTSPICE Comparisons of Schematic & Layout pg. 99**

**Section 11) Conclusion pg. 100**

**Section 12) References and Electric Files pg. 101**

**Section 1) Objective**

For this lab, the objective will be to apply everything we’ve learned about Quartus II in the tutorials. I will be doing the following:

* Building the following circuits using Object form and VHDL: 2to1 Multiplexer, 1-bit Half Adder, 1-bit Full Adder, 3to8 Decoder and 8to3 Encoder
* Verifying their correctness using waveform simulations
* Writing testbench files in VHDL to test the correct of the designs
* Programming pin assignments for the board

**Section 2) Description and Specifications**

2to1 Multiplexer

The first circuit I will be designing is a 2to1 **Multiplexer**. A multiplexer, also known as a mux, is “basically a switch that passes one of its data inputs through to the output, as a function of a set of select inputs”[1]. One of their uses is to choose among several multibit input numbers. The typical logic symbol of a 2to1 Multiplexer is shown below in Figure **NUM HERE**

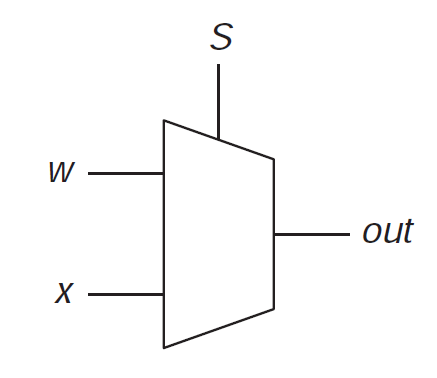


Figure NUM: 2to1 Multiplexer

The way a 2to1 multiplexer works as follows. If the select input, *S*, is equal to 0, the output, *out*, is equal to the value of X. If the select input is equal to 1, the output is equal to the value of Y. Table **NUM HERE** below shows the truth table of a 2to1 Mux. I will denote the two inputs as X and Y and the output as M.

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **Y** | **S** | **M** |
| **0** | **0** | **0** | **0** |
| **0** | **0** | **1** | **0** |
| **0** | **1** | **0** | **0** |
| **0** | **1** | **1** | **1** |
| **1** | **0** | **0** | **1** |
| **1** | **0** | **1** | **0** |
| **1** | **1** | **0** | **1** |
| **1** | **1** | **1** | **1** |

Table 1: 2to1 Multiplexer Truth Table

A simplified table is shown in Table **NUM HERE** below

|  |  |
| --- | --- |
| **S** | **M** |
| **0** | **X** |
| **1** | **Y** |

Table 2: 2to1 Multiplexer - Simplified Truth Table

We can derive the Boolean algebra expression of a 2to1 multiplexer from table **NUM BERE**. Looking at the table, we get the following

Equation 1: Out of 2to1 Mux

This can be composed of two AND gates, one OR gate and one NOT gate. The design of AND, OR and NOT gates can be designed using transistors, but are not within the scope of this course, so shall not be discussed.

The inputs and outputs will be assigned as follows on our board, seen in Figure **NUM HERE** below. It comes from the pin assignment text file for this circuit.

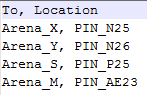


Figure 1: Pin Assignment for 2to1 Mux

The format is as follows. To, Location. To is the input/outputs from the object/vhdl file. The Location is the appropriate pins used for inputs and outputs. The pins are gotten from the **INSErT SOURCE HERE**.

On the next page in Figure **NUM HERE** is the design I made in Quartus for the 2to1 Multiplexer.

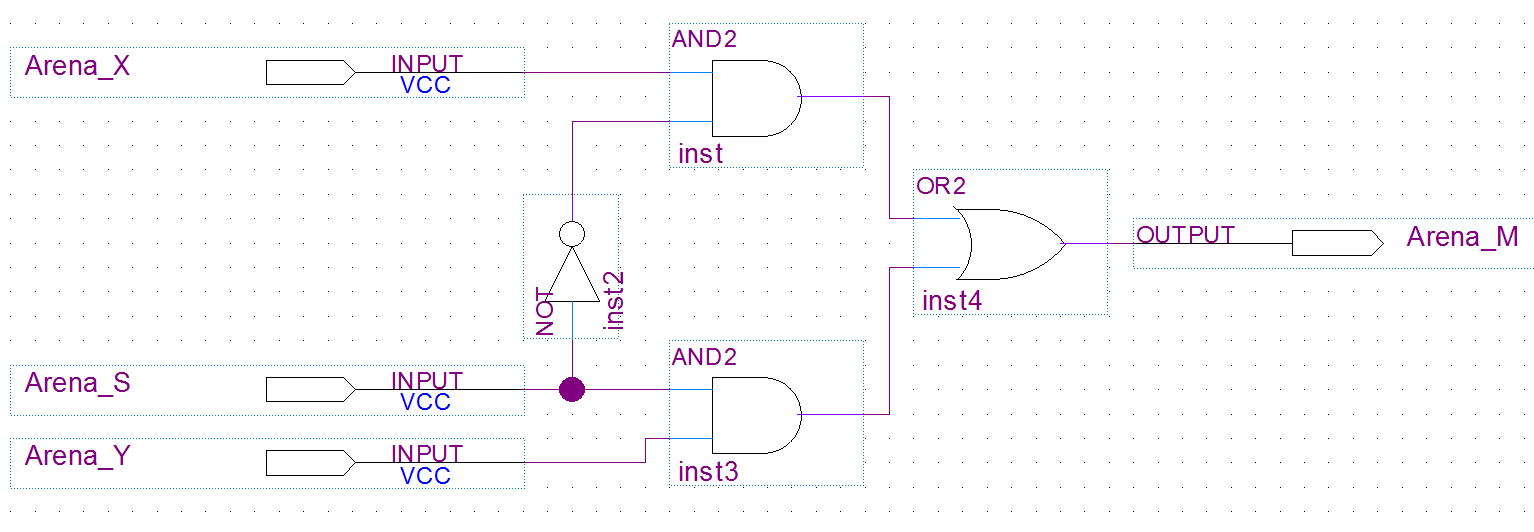


Figure 2: 2to1 Multiplexer at the Logic Level

As can be seen in the figure, the output consists of an 2-input OR gate connected to two 2-input AND gates with the appropriate inputs, with one AND gate having it’s input connected to a NOT gate just as described in Equation 1.

Below in Figure **NUM HERE** is the VHDL code for the circuit.

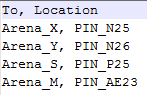


Figure 3 : VHDL Code for 2to1 Mux

1-bit Half Adder

The second circuit I will be designing is a **1-bit Half Adder.** A 1-bit Half Adder is a circuit that can add two 1-bit numbers, and produce a Sum and a possible carry over. We know from basic math for example, if we had 3+3, we will have a sum of 6 with a carry over of 0, but 7+4 will produce a sum off 11 with a carry over of 1. We know from Boolean algebra that a 1-bit number is base two, since it can only have two digits, either a 0 or a 1. Since that is the case, let’s quickly review the rules of binary addition of 1-bit numbers

Knowing these rules, below in Table **NUM** describes the functionality of a 1-bit half adder.

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **Y** | **Sum** | **Carry Out** |
| **0** | **0** | **0** | **0** |
| **0** | **1** | **1** | **0** |
| **1** | **0** | **1** | **0** |
| **1** | **1** | **0** | **1** |